Vertex Classification on Weighted Networks

by

Hayden Helm

A thesis submitted to Johns Hopkins University
in conformity with the requirements for the degree of
Master of Science in Engineering

Baltimore, Maryland
December, 2018

Abstract

This paper provides a framework for vertex classification on weighted networks. We assume that the edge weights and adjacencies in the network are conditionally independent and that both sources of information encode class membership information. In particular, we introduce a edge weight distribution matrix to the standard K-Block Stochastic Block Model to model weighted networks. This allows us to develop simple yet powerful extensions of classification techniques using the spectral embedding of the unweighted adjacency matrix. In this paper we look at two settings for the edge weight distributions and propose classification procedures in both settings. We show the effectiveness of the proposed classifiers by comparing them to pass-to-ranks. Moreover, we discuss and show how our method performs when the edge weights do not encode class membership information.

Acknowledgments

I'd like to thank Dr. Carey E. Priebe for supporting the work presented in this paper and for providing a fun and productive research environment. More importantly, I'd also like to thank him for his contagious enthusiasm for mathematics and for life in general.

I'd like to thank the entire JHU Applied Math and Statistics department for providing a challenging yet rewarding undergraduate experience.

I'd like to thank my family – blood and otherwise – for supporting me as I pursue my dreams. They all constantly inspire and challenge me to be the best person I can be. My accomplishments would not be possible without the foundation of love and support they all provide.

I'd like to thank my girlfriend, Anna, for her editing expertise throughout writing this paper and for her unwavering kindness and patience in general. She's awesome.

Finally, I'd like to thank my Mom for always believing in me and for showing me what it means to persevere. I have not discovered the "pint" yet, but I hope this paper is a good alternative in the meantime.

Table of Contents

List of Figures

Chapter 1

Introduction

Weighted networks are common in many research fields ranging from neuroscience to sociology. While networks provide a rich source of information, it can be difficult to identify patterns and groupings within the data. Hence, problems that require understanding relationships within and across groups of nodes, which we will refer to as communities or classes, are non-trivial. For vertex, or node, classification the objective is to predict the class label for each node where we assume that nodes belong to exactly one of K classes.

Pass- to-ranks is an existing method for node classification on weighted networks. It uses the rankings of the edge weights to transform a weighted adjacency matrix from $A \in \mathbb{R}^{nxn}$ to $A_{ptr} \in ([0,2] \cap \mathbb{Q})^{nxn}$ by changing the value of the edge weight. The new weight is equal to two times the rank of the original edge of A divided by $|E|$, the size of the edge set. While pass-to-ranks is useful for node classification, it can be hard to pin down analytically due to the method's minimal assumptions on the edge weights. One of the goals of the this paper is to provide a more tractable framework for effective node classification on weighted networks. A nice overview of vertex classification

techniques can be found in (Bhagat, Cormode, and Muthukrishnan, 2011).

Following this introduction, section 2 provides necessary preliminary information. Sections 3 and 4 present new methods for node classification in a weighted network, showcase the effectiveness of the proposed methods, and discuss their sensitivity to misspecification. In section 3 we classify in a setting where it is assumed that the weight distributions are ordered. In section 4 we classify in a more general setting. Afterwards, section 5 discusses method limitations and areas for further study. Section 6 concludes with final remarks.

1.1 Problem Statement

It is important to completely characterize the problem we address before we continue. We use notation and concepts here that are explained in more detail in later sections.

In our setting our goal is to classify unlabeled nodes in a weighted network. In general, we are given a weighted network, $G = (V, E)$, where V is a set of nodes and E is a set of edges. Note that $(i, j, w_{ij}) \in E$ if the edge between node i and node j exists and has weight $w_{i,j}$. In our setting we deal with symmetric (if $(i, j, w_{i,j}) \in E$ then $(j, i, w_{i,j}) \in E$) and hollow $(i, i, w_{i,i}) \notin E$) networks.

We represent this network as a weighted adjacency matrix, denoted C, where we can think of C as the Hadamard, or entrywise, product of the unweighted adjacency matrix A and the matrix of weights W. That is, $C = A \odot W$. Additionally, we are given a set of nodes with known class membership, referred to as training or labeled nodes, that we can use to inform our procedure. In this paper there's an explicit assumption that W encodes block membership

information. There exists powerful methods for dealing with A, outlined in section 2.3, and so our focus will be on handling W and, in turn, C.

This paper is an exploration of how we can use the class membership information encoded in W to more accurately classify unlabeled nodes. Specifically, we assume that there is a symmetric, $K \times K$ matrix of distributions, $\mathcal{F}$, where the $(u, v)^{th}$ entry of $\mathcal{F}$ is the distribution governing the edge weights between the nodes in block u and the nodes in block v (sections 2.2 and 2.4.1). We estimate these distributions using the edge weights between the training nodes in block u and the training nodes in block v. The estimated distributions are denoted $\hat{\mathcal{F}}_{u,v}$. Consequently, for block u, we have a vector of estimated distributions $\hat{\mathcal{F}}_u = (\hat{F}_{u,1}, \ldots, \hat{F}_{u,K})$.

Our focus now turns to a single unlabeled node. In this setting we observe the edge weights between an unlabeled node and the training nodes for each block. Hence, for a particular unlabeled node i we can estimate the distributions corresponding to each block. That is, $\hat{\mathcal{F}}(i) = (\hat{\mathcal{F}}(i)_1, \ldots, \hat{\mathcal{F}}(i)_K)$ for unlabeled node i. Extracting class membership information for the unlabeled node from this collection of vectors comes down to comparing $\hat{\mathcal{F}}(i)$ to each of the $\hat{\mathcal{F}}_u$.

Letting $\hat{\mathcal{F}}_{u,v}$ be the empirical cumulative distribution is perhaps the most general treatment of the edge weight distributions and is addressed in the analysis below. We also explore different assumptions on the distributions and propose classification procedures for these settings.

Chapter 2

Preliminaries

2.1 Stochastic Block Model

The network model used in this paper is the Stochastic Block Model (SBM), which is a restricted version of the Random Dot Product Graph (RDPG) (Young and Scheinerman, 2007). An RDPG is an independent edge random graph that is characterized by a collection of positions in $\mathbb{R}^d$ that correspond to the nodes in the network. In particular, each node i in the network has a "position", $X_i \in \mathbb{R}^d$ where the only restriction on X_i is that $\langle X_i, X_j \rangle \in [0,1]$ for all i, j, where $\langle \cdot, \cdot \rangle$ is the dot product of two vectors. The SBM is an RDPG where $X_i \in \{X_1, \ldots, X_K\}$, where K is the number of blocks or classes.

In an SBM the probability that an edge exists between two nodes depends only on the class memberships of the nodes. Importantly, the true positions are typically unknown and are referred to as latent positions. We call the estimates of the latent positions estimated positions.

The SBM is a common generative model used for network analysis because of its simple description and ability to capture complex network structures (see

(Abbe, 2017) sections 1 and 2 for history and literature overview). Four objects completely describe the model. The number of blocks in the network, K. The set of nodes, V, where $|V| = n$. The (sometimes partially observed) block membership function $b : V \to [K]$ which implies block membership priors $\pi = (\pi_1, \pi_2, .., \pi_K) \in \Delta_K$. And, finally, the matrix that governs adjacency information

$$B = \begin{bmatrix} \langle X_1, X_1 \rangle & \dots & \langle X_1, X_K \rangle \\ \vdots & \ddots & \vdots \\ \langle X_K, X_1 \rangle & \dots & \langle X_K, X_K \rangle \end{bmatrix}$$

where X_u is the latent position corresponding to nodes in block u. The existence of an edge between node i and node j, where $b(i) = u$ and $b(j) = v$, is generated from a coin flip with weight equal to $B[b(i), b(j)] = B_{u,v}$.

The analysis in this paper is focused on 2 block matrices of the form

$$B = \begin{bmatrix} p^2 & pq \\ pq & q^2 \end{bmatrix}$$

Notice that $det(B) = p^2q^2 - p^2q^2 = 0$. Using the characteristic equation to find the eigenvalues,

$$\lambda^2 - (p^2 + q^2)\lambda = 0$$

$$\lambda(\lambda - p^2 - q^2) = 0$$

$$\lambda_1, \lambda_2 = 0, p^2 + q^2$$

So $rank(B) = 1$ if $p > 0$ or $q > 0$. We assume that $rank(B) = 1$ is known throughout this paper. Otherwise, estimating $rank(B)$ is a complicated task in and of itself (Zhu and Ghodsi, 2006).

2.2 Adjacency Spectral Embedding

An important method used in the present paper is the Adjacency Spectral Embedding (ASE) of a network. ASE transforms the network into a collection of objects in Euclidian space using the Singular Value Decomposition (SVD). In particular,

$$A = U\Sigma V^T$$

where U and V are orthogonal and Σ is a diagonal matrix with the the singular values of A occupying the diagonals in decreasing order. (Athreya et al., 2016) shows that if A is generated from an RDPG then the rows of $U\Sigma^{1/2}$ are asympotically normally distributed around an orthogonal transformation of the latent positions that generated B. Moreover, (Sussman, Tang, and Priebe, 2014) shows that the error rate for classification procedures that use k-nearest neighbors on the spectral embedding of the adjacency matrix converge in probability to Bayes' error. Clearly, the adjacency spectral embedding is of practical use. See (Von Luxburg, 2007) for an implementation tutorial.

For the two block rank one case

$$X_i \sim \mathcal{N}\left(p, \frac{\pi_1 p^4(1-p^2) + \pi_2 pq^3(1-pq)}{n(\pi_1 p^2 + \pi_2 q^2)^2}\right) \text{ if } b(i) = 1$$

$$X_i \sim \mathcal{N}\left(q, \frac{\pi_1 p^3 q(1-pq) + \pi_2 q^4(1-q^2)}{n(\pi_1 p^2 + \pi_2 q^2)^2}\right) \text{ if } b(i) = 2$$

Thus, modeling the spectral embedding of the adjacency matrix as a mixture of Gaussians is not only analytically convenient but also theoretically sound.

See (Athreya et al., 2017) for a survey of results on spectral embeddings of RDPGs.

2.3 Pattern Recognition

Classification tasks require labeling objects whose group membership is unknown. Generally, we can consider a classifier as a function from an input space $\mathcal{X}$ to a set of labels. Namely, $h : \mathcal{X} \to [K]$. In the current setting we consider $\mathbb{R}$ and $\mathbb{R}^d$ as input spaces. For objects in $\mathbb{R}^d$, it is intuitively appealing to think of the entire space as "painted" by K colors, with $X \in \mathbb{R}^d$ colored k if $h(X) = k$. $h(\cdot)$ is typically unknown and there are numerous methods for estimating it. This paper focuses exclusively on Bayes' classifier. See (Fishkind et al., 2015) for pattern recognition schemes for unweighted networks. Consider the following simple example.

In a local high school thirty percent of the students are athletes. For a given athlete, their resting heart rate is a random variable from a continuous distribution defined on the positive real numbers with density denoted $f_A(\cdot;\theta_A)$. An analogous density defines the resting heart rate for non-athletes, $f_N(\cdot;\theta_{NA})$. Let an unlabeled student have a resting heart rate h. Since our goal is to classify the unknown student, it is natural to compare how likely h came from the distribution governing athletes to how likely h came from the distribution governing non-athletes. That is, if the two densities are known then the student can be classified by comparing $f_A(h;\theta_A)$ to $f_N(h;\theta_N)$ with the appropriate weights. That is, classify the unknown student as an athlete if $(0.3)f_A(h;\theta_A) > (0.7)f_N(h;\theta_N)$ and as a non-athlete otherwise. This intuitive

approach to classification is exactly a comparison of the product of conditional likelihoods and priors and is aptly called the Bayes classifier. In a general setting with K groups, the Bayes classifier will classify an unlabeled object x as a member of group j if

$$g(x) = \arg\max_{i \in [K]} \pi_i f_i(x; \theta_i) = j$$

The parameters that determine the distributions are usually unknown and need to be estimated. If $\theta \in \Theta$ is estimated by $\hat{\theta}$ and $f(\cdot; \theta)$ is estimated by $f(\cdot; \hat{\theta})$ then the classifier that uses the estimated densities is called a plug-in classifier.

2.3.1 Univariate Normal, two class Bayes classifier

Consider a two-class classification problem in $\mathbb{R}$ where the generative distributions are known to be Gaussian. Furthermore, suppose that the means and variances of the two distributions are known. WOLOG suppose $\mu_1 < \mu_2$. Then the Bayes decision boundary is given by

$$\frac{\pi_1}{\sigma_1} \exp\left\{\frac{-1}{2\sigma_1^2}(x_i - \mu_1)^2\right\} = \frac{\pi_2}{\sigma_2} \exp\left\{\frac{-1}{2\sigma_2^2}(x_i - \mu_2)^2\right\}$$

$$\exp\left\{\frac{1}{2\sigma_2^2}(x_i - \mu_2)^2 - \frac{1}{2\sigma_1^2}(x_i - \mu_1)^2\right\} = \frac{\pi_2 \sigma_1}{\pi_1 \sigma_2}$$

$$\frac{1}{2\sigma_2^2}(x_i - \mu_2)^2 - \frac{1}{2\sigma_1^2}(x_i - \mu_1)^2 = \log\left(\frac{\pi_2 \sigma_1}{\pi_1 \sigma_2}\right)$$

$$\sigma_1^2(x_i - \mu_2)^2 - \sigma_2^2(x_i - \mu_1)^2 = 2\sigma_1^2\sigma_2^2 \log\left(\frac{\pi_2 \sigma_1}{\pi_1 \sigma_2}\right)$$

and, finally,

$$(\sigma_1^2 - \sigma_2^2)x_i^2 + 2(\mu_1\sigma_2^2 - \mu_2\sigma_1^2)x_i - \mu_1^2\sigma_2^2 + \mu_2^2\sigma_1^2 + 2\sigma_1^2\sigma_2^2 \log\left(\frac{\pi_1\sigma_2}{\pi_2\sigma_1}\right) = 0$$

which yields two solutions, denoted $x_\pm$:

$$x_\pm^* = \frac{\mu_2\sigma_1^2 - \mu_1\sigma_2^2 \pm \sqrt{\left(\mu_1\sigma_2^2 - \mu_2\sigma_1^2\right)^2 - \left(\sigma_1^2 - \sigma_2^2\right)\left(\mu_2^2\sigma_1^2 - \mu_1^2\sigma_2^2 + 2\sigma_1^2\sigma_2^2 \log(\frac{\pi_1\sigma_2}{\pi_2\sigma_1})\right)}}{\sigma_1^2 - \sigma_2^2}$$

We highlight the uni-variate case because our results are generated with a rank one SBM and so the spectral embedding of the adjacency matrix is univariate. The algebra exercise above is simply to build an intuition as to what the classifier we propose is actually doing. Moreover, by understanding where the decision boundaries come from we can shift them by tuning parameters.

2.4 A small example (part 1)

Consider C, the weighted, hollow and symmetric adjacency matrix that is generated from an SBM with parameters $n = 10$ and $B = \begin{bmatrix} (0.8)^2 & (0.8)(0.6) \\ (0.8)(0.6) & (0.6)^2 \end{bmatrix}$. Suppose we know the class memberships of nodes 1, 2, 6 and 7. Namely,

$b(1) = b(2) = 1$ and $b(6) = b(7) = 2$.

$$C = \begin{bmatrix} 0 & 2 & 2 & 0 & 2 & 2 & 0 & 0 & 0 & 1 \\ 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 0 & 2 & 0 & 0 & 3 & 4 & 0 & 4 \\ 0 & 0 & 2 & 0 & 1 & 3 & 2 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 & 3 & 0 & 0 & 0 \\ 2 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 2 & 3 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 6 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 6 & 0 & 0 \\ 1 & 0 & 4 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \end{bmatrix}$$

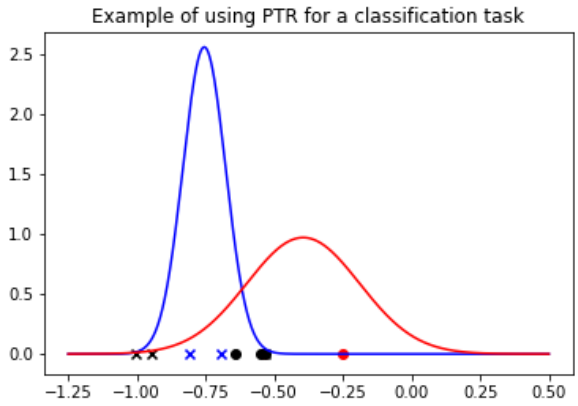


Figure 2.1: An illustration of how to use pass-to-ranks for a classification task. The blue curve is the estimated Gaussian for block 1 and the red curve is the estimated Gaussian for block 2. The nodes from block 1 are x's and the nodes from block 2 are o's. Unlabeled nodes are black.

If we want to apply pass-to-ranks to C we first count the number edges (17 – our network is undirected) and give each nonzero edge weight a rank. For the sake of clarity we will consider $ptr(C) : \mathbb{R}^{n \times n} \to [0,1]^{n \times n}$. There is one 6, so we give it rank $|E| = 17$. There are three 4s, so we give them rank

$\frac{3|E|-(1+2+3)}{3} = 15$, and so on. Resulting in

$$ptr(C) = C_{ptr} = \begin{bmatrix} 0 & \frac{6.5}{17} & \frac{6.5}{17} & 0 & \frac{6.5}{17} & \frac{6.5}{17} & 0 & 0 & 0 & \frac{1.5}{17} \\ \frac{6.5}{17} & 0 & \frac{6.5}{17} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{6.5}{17} & \frac{6.5}{17} & 0 & \frac{6.5}{17} & 0 & 0 & \frac{12}{17} & \frac{15}{17} & 0 & \frac{15}{17} \\ 0 & 0 & \frac{6.5}{17} & 0 & \frac{1.5}{17} & \frac{12}{17} & \frac{6.5}{17} & 0 & 0 & 0 \\ \frac{6.5}{17} & 0 & 0 & \frac{1.5}{17} & 0 & 0 & \frac{12}{17} & 0 & 0 & 0 \\ \frac{6.5}{17} & 0 & 0 & \frac{12}{17} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{12}{17} & \frac{6.5}{17} & \frac{12}{17} & 0 & 0 & 0 & \frac{15}{17} & 0 \\ 0 & 0 & \frac{15}{17} & 0 & 0 & 0 & 0 & 0 & 1 & \frac{6.5}{17} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{15}{17} & 1 & 0 & 0 \\ \frac{1.5}{17} & 0 & \frac{15}{17} & 0 & 0 & 0 & 0 & \frac{6.5}{17} & 0 & 0 \end{bmatrix}$$

We then find the Singular Value Decomposition of $C_{ptr} = U\Sigma U^T$ and take the first column of $\hat{X} = U\Sigma^{1/2}$ as the estimated position for each node. That is, the latent positions can be estimated by

$$\hat{X} = [-0.81, -0.70, -1.00, -0.94, -0.53, -0.25, -0.54, -0.64, -0.55, -0.53]^T$$

Under the assumption that the latent positions are distributed normally, we can estimate the parameters of the Gaussian mixture model, with $\hat{\mu_1} = \frac{-0.81-0.70}{2} = -.755$, $\hat{\sigma_1} = 0.078$, $\hat{\mu_2} = \frac{-0.25+-0.54}{2} = -0.395$, $\hat{\sigma_2} = 0.205$, resulting in the distributions in Figure 2.1. Finally, we can classify an unlabeled node based on the likelihood of observing its estimated position under each Gaussian. For example, the likelihood of observing the estimated position corresponding to node 3 under block 1 is around 0.03. The likelihood under block 2 is around 0.02. Therefore, we classify node 3 as a member of block 1. We will return to this example again later.

2.4.1 A second perspective

Consider a weighted, symmetric and hollow matrix $C \in \mathbb{R}^{n \times n}$. Recall from section 1.1 that we can think of this matrix as the Hadamard product (denoted $\odot$) of $A \in \{0,1\}^{n \times n}$ and $W = \mathbb{R}^{n \times n}$ where $a_{i,j}$ is 1 if there is an edge between node i and node j and 0 otherwise. $w_{i,j}$ is the weight of the edge between node i and node j. Using the C in section 2.4 as an example,

$$C = \begin{bmatrix} 0 & 2 & 2 & 0 & 2 & 2 & 0 & 0 & 0 & 1 \\ 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 0 & 2 & 0 & 0 & 3 & 4 & 0 & 4 \\ 0 & 0 & 2 & 0 & 1 & 3 & 2 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 & 3 & 0 & 0 & 0 \\ 2 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 2 & 3 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 6 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 6 & 0 & 0 \\ 1 & 0 & 4 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\odot \begin{bmatrix} x & 2 & 2 & x & 2 & 2 & x & x & x & 1 \\ 2 & x & 2 & x & x & x & x & x & x & x \\ 2 & 2 & x & 2 & x & x & 3 & 4 & x & 4 \\ x & x & 2 & x & 1 & 3 & 2 & x & x & x \\ 2 & x & x & 1 & x & x & 3 & x & x & x \\ 2 & x & x & 3 & x & x & x & x & x & x \\ x & x & 3 & 2 & 3 & x & x & x & 4 & x \\ x & x & 4 & x & x & x & x & x & 6 & 2 \\ x & x & x & x & x & x & 4 & 6 & x & x \\ 1 & x & 4 & x & x & x & x & 2 & x & x \end{bmatrix}$$

where x is an unobserved weight between node i and node j. Note that each x should be indexed by i and j but this dependence is suppressed to avoid cluttering the matrix.

Splitting C into A and W gives us a reason to consider adding another component to the SBM to address W. With this thought in mind, it seems natural to propose a matrix of distributions, $\mathcal{F}$, where $\mathcal{F}_{u,v}$, $u, v \in [K]$ is the distribution governing the edge weights between block u and block v. It is clear that $\mathcal{F}$ is analogous to the matrix B in the standard SBM, which models the adjacency relationship between nodes in block u and nodes in block v. While this extension of the SBM is completely natural, the question remains about how to use this additional component for classification tasks.

Chapter 3

Ordered Edge Weight Distributions

Walking through the procedure in section 2.4 gives some insight as to why pass-to-ranks is an effective method. It combines the adjacency and edge weight information in a way such that neither dominates the other. However, the usefulness of the weight information depends on there being an ordered relationship that can be captured by a simple ranking mechanism.

Assuming that the weights of the network encode information about block membership, if we use pass-to-ranks we'd hope that the edge weights between nodes in block u and nodes in block v have some ordered relationship, i.e. $\mathbb{E}(w_{u,v}) < \mathbb{E}(w_{u,t})$ for $u, v, t \in [K]$. That is, the edge weights between nodes in block u and block v come from a distribution with a different mean than the edge weights between nodes in block u and block t.

While we do not know order of the distributions, the partially observed $b(\cdot)$ allows us to estimate the ordering using the weights between training data. For each unlabeled node we can estimate the ordering of its distributions using the edge weights between it and the training data. These estimated orderings can be used as proxies for $\hat{\mathcal{F}}_u$ and $\hat{\mathcal{F}}(i)$ from section 1.1.

In this section we expand on the idea of ranking objects as a similarity metric by comparing the estimated ordering for an unlabeled node to the estimated ordering for each block. We use the results of this comparison to update the class membership priors for each unlabeled node. We demonstrate the effectiveness of this method as compared to pass-to-ranks for data that is generated from an SBM with the additional weight distribution component $\mathcal{F}$. In an example in section 3.3 we use the footrule distance on a pair of permutations to find the dissimilarity between them. The footrule distance is the sum of absolute differences of the indices of the set of objects. That is, if we have two permutations of $[4]$, $P_1 = (1, 2, 3, 4)$ and $P_2 = (2, 3, 4, 1)$ then $d_{FR}(P_1, P_2) = \sum_{i=1}^{4} |\arg_i(P_1) - \arg_i(P_2)| = |1 - 4| + |2 - 1| + |3 - 2| + |4 - 3| = 6$, where $\arg_i(P_j)$ returns the index of the i in P_j.

3.1 Model Assumptions

We assume that the network is generated from a K-Block SBM with partially observed block membership function $b(\cdot)$, unobserved B and unobserved $\mathcal{F}$. Moreover, we assume that the edge weight distributions have finite expectation and that $E(F_{u,v}) \neq E(F_{s,t})$ for all $u, v, s, t \in [K]$. It is then possible to order the distributions based on expected value, i.e. there exists an ordering such that $E(F_{(1)}) < \ldots < E(F_{(L)})$ where $L = \binom{K}{2} + K$ and $F_{(i)}$ is the i^{th} ranked distribution.

Let $O(\mathcal{F}) = (E(F_{(1)}), \ldots, E(F_{(L)}))$ be the ordering of the distributions for a specific K-block SBM with weight distribution matrix $\mathcal{F}$ and the appropriate restrictions on $F_{u,v}$. We sometimes refer to $O(\mathcal{F})$ as the "global" ordering.

The global ordering implies a collection of K "local" orderings, $O_u(\mathcal{F}) = (F_{u,(1)}, \ldots, F_{u,(K)})$ for $u \in [K]$. Moreover, each node i in V has an associated ordering based on block membership, i.e. $\bar{O}_i(\mathcal{F}) = (F_{i,(1)}, \ldots, F_{i,(K)}) = O_{b(i)}(\mathcal{F})$. We view O_u and $\bar{O}_i$ as proxies for the vectors of estimated distributions from section 1.1. That is, instead of comparing vectors of estimated distributions directly we can compare different permutations of $[K]$.

3.2 Methodology

As discussed previously and showcased in section 2.4, classification tasks for graph objects can be done via spectral methods and, in particular, using the spectral embedding of the adjacency matrix of the graph. Once the spectral embedding is obtained, any method used for Euclidian data can be applied to the estimated positions. A mixture of Gaussians is used for this method, with parameter estimations based on the training data. The partially observed block membership function, for example, can be used to estimate the block membership prior associated with each Gaussian in the mixture. That is, $\hat{\pi} = (\hat{\pi}_1, \ldots, \hat{\pi}_K)$ where

$$\hat{\pi}_u = \frac{\sum_{i \in N_u} \mathbb{1}_{\{b(i)=u\}}}{\sum_{v \in [K]} |N_v|}$$

for all $u \in [K]$, where N_u is the set of training nodes for block u.

One way to use the block membership information encoded in $\mathcal{F}$ is to integrate it into tried and trusted procedures. There are a few things to consider when doing this. Firstly, we know that fitting a mixture of Gaussians using the spectral embedding of the unweighted adjacency matrix works well for

clustering tasks on unweighted networks. Secondly, our shift of perspective (section 2.4.1) and the addition of $\mathcal{F}$ means there is more information about class membership available. Hence, unless we wish to deviate from spectral based methods, we must use the additional block membership information to update our block membership priors.

We define a permutation error for each ordering and convert the error into a measure of similarity that is consequently used to update the prior for each unlabeled node. There are innumerable dissimilarities on permutations to consider – footrule distance, Kendall's Tau, 0-1 error, etc. We let $d(\cdot,\cdot)$ be the dissimilarity metric and let $d_{i,u}$ be the dissimilarity between O_u and $\bar{O}_i$. Namely, $d_{i,u} = d(\bar{O}_i, O_u)$. We then define $D_i = (d_{i,1}, \ldots, d_{i,K})$ as the error vector for unlabeled node i. Normalizing the error vector results in

$$ND_i = \frac{1}{\sum_{u\in[K]} d_{i,u}} D_i$$

and we define the similarity vector to be

$$S_i = \hat{1}_K - (ND_{i,1}, \ldots, ND_{i,K})$$

where $\hat{1}_K$ is the vector of ones of length K. Finally, we update our priors

$$\hat{\pi}_i = \frac{1}{\langle \hat{\pi}, S_i \rangle} (\hat{\pi}_1 S_{i,1}, \ldots, \hat{\pi}_K S_{i,K}) = (\hat{\pi}_{i,1}, \ldots, \hat{\pi}_{i,K})$$

where $\langle \cdot,\cdot \rangle$ is the inner product of two vectors. Then our classifier is

$$g_R(x_i) = \arg\max_{u\in[K]} \hat{\pi}_{i,u} f(x_i; \hat{\theta}_u)$$

where $f(\cdot; \hat{\theta}_u)$ is the estimated Gaussian density for block u. Notice that this

new classifier utilizes both adjacency and weight information – in short, we have successfully integrated the new class membership information.

We recognize that we reuse notation when defining the estimated class membership priors found using $b(\cdot)$ and the updated class priors. The meaning of $\hat{\pi}_i$ or $\hat{\pi}_u$ should be clear in context – one refers to the updated prior vector for node i and the other refers to the original estimated class membership prior for block u.

3.3 Properties of updating priors in the two block case

The two block rank one case sheds light on the mechanics of the methodology. First recall the decision boundaries from section 2.3.1:

$$x^*_{\pm} = \frac{\mu_2\sigma_1^2 - \mu_1\sigma_2^2 \pm \sqrt{\left(\mu_1\sigma_2^2 - \mu_2\sigma_1^2\right)^2 - \left(\sigma_1^2 - \sigma_2^2\right)\left(\mu_2^2\sigma_1^2 - \mu_1^2\sigma_2^2 + 2\sigma_1^2\sigma_2^2\log(\frac{\pi_1\sigma_2}{\pi_2\sigma_1})\right)}}{\sigma_1^2 - \sigma_2^2}$$

Tuning the ratio of the block membership priors has an explicit effect on the position of the decision boundaries. Information on the ordering of the distributions allows us to move this boundary in a non-arbitrary way for each unlabeled node.

Note that updating priors with the same error, and thus the same similarity, would result in an "update" of the priors for vertex i such that $\hat{\pi} = \hat{\pi}_i$, i.e. the "updated" prior for vertex i would be the same as the prior estimated from the observed portion of $b(\cdot)$ (see example below). Thus, we can focus our attention on the case where a disagreement occurs.

Without loss of generality, assume $O_1 = (1, 2)$ and $O_2 = (2, 1)$. Then for an unlabeled node i, $\bar{O}_i$ is equal to O_1 or O_2. To illustrate the mechanics of the

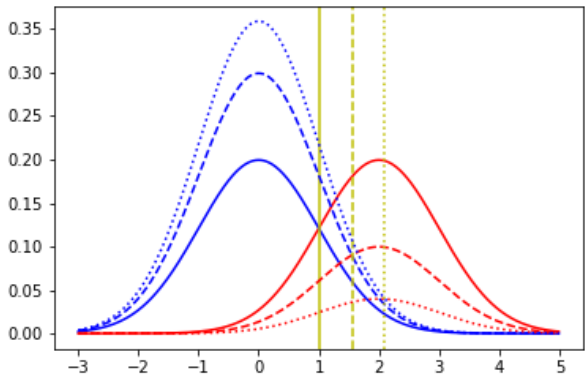


Figure 3.1: An illustration of how changing the priors associated with each Gaussian moves the decision boundary for $\hat{\mu}_1 = 0, \hat{\mu}_2 = 2, \hat{\sigma} = 1, \hat{\pi}_1 = \{0.5 \text{ (solid)}, 0.75 \text{ (dashed)}, 0.9 \text{ (dotted)}\}$.

method, let $\bar{O}_i = O_1$. When we include a base error of one, discussed in detail below, and use the footrule distance, we obtain a normalized error vector $ND_i = (\frac{1}{4}, \frac{3}{4})$ and corresponding similarity vector $S_i = \hat{1}_2 - ND_i = (\frac{3}{4}, \frac{1}{4})$, resulting in

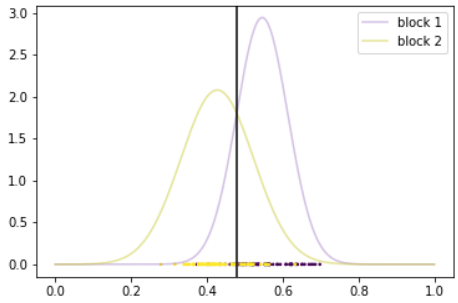
$$\hat{\pi}_i = \frac{1}{\frac{3}{4}\hat{\pi}_1 + \frac{1}{4}\hat{\pi}_2}(\frac{3}{4}\hat{\pi}_1, \frac{1}{4}\hat{\pi}_2)$$

which leads to new decision boundaries

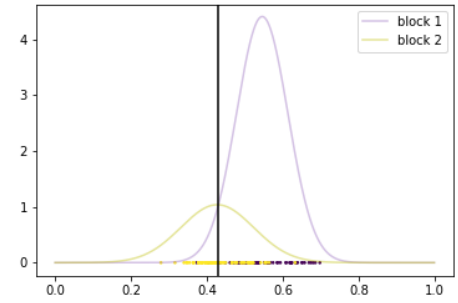
$$x^*_{\pm,i} = \frac{\mu_2\sigma_1^2 - \mu_1\sigma_2^2 \pm \sqrt{(\mu_1\sigma_2^2 - \mu_2\sigma_1^2)^2 - (\sigma_1^2 - \sigma_2^2)(\mu_2^2\sigma_1^2 - \mu_1^2\sigma_2^2 + 2\sigma_1^2\sigma_2^2 \log(\frac{\hat{\pi}_{i,1}\sigma_2}{\hat{\pi}_{i,2}\sigma_1}))}}{\sigma_1^2 - \sigma_2^2}$$

dependent on the particular unlabeled node.

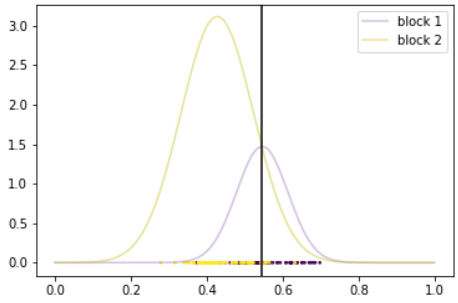
Figure 3.2 shows how updating priors changes the decision boundary for particular nodes in the setting where $B = \begin{bmatrix} (0.55)^2 & (0.55)(0.45) \\ (0.55)(0.45) & (0.45)^2 \end{bmatrix}$, $\pi_1 = \pi_2 = 0.5$, $n = 150$, $\mathcal{F} = \begin{bmatrix} N(5,1) & N(10,1) \\ N(10,1) & N(5,1) \end{bmatrix}$. From the figure we can see that an informed shift in the decision boundary can have a huge impact on classification results. For example, the right most plot in Figure 3.2



(a)



(b)



(c)

Figure 3.2: a) shows the densities and corresponding decision boundary associated with the original priors. b) shows the densities and corresponding decision boundary when an unlabeled node's ordering matches with the ordering for block 1. c) shows the densities and corresponding decision boundary when an unlabeled node's ordering matches the ordering for block 2. Please note that there are actually two decision boundaries for each case (since the variances of the Gaussians are typically unequal) but we have muted the less impactful one.

would correctly classify more unlabeled nodes whose latent block is block 2 (yellow) than the original classifier. Again, moving the decision boundary in an informed way can decrease misclassification rates. Simulation results are discussed thoroughly in section 3.5.

The base error of one that we applied is called plus-one smoothing and is generally used to avoid method or model degradation, see Gale and Church, 1994. In our case, if we did not apply it we would classify unlabeled nodes solely on the information contained in the edge weights. A simple way to see this is in the example we presented above. Recall that $O_1 = \bar{O}_i = (1, 2)$ and $O_2 = (2, 1)$. If we did not apply plus-one smoothing we would end up with the error vector $(0, 2)$, which yields the normalized error vector $(0, 1)$ and the similarity vector $(1, 0)$. The updated prior would be $(1, 0)$ and we would completely ignore all other block membership information when classifying.

Interestingly, additive smoothing can have a significant impact on our procedure. Imagine that instead of plus-one, we used plus-10000 smoothing. Then, doing the same as before, we get the error vector $(10000, 10002)$, the normalized error vector $(\frac{10000}{20002}, \frac{10002}{20002}) \approx (0.5, 0.5)$ and, finally, the similarity vector that is approximately $(0.5, 0.5)$. But this similarity vector gives us no additional class membership information! In fact with the current method, plus-10000 smoothing would spit out approximately our original priors. In section 3.6 we look at a dynamic type of additive smoothing that is less naive than plus-one smoothing and less rigid than plus-10000 smoothing.

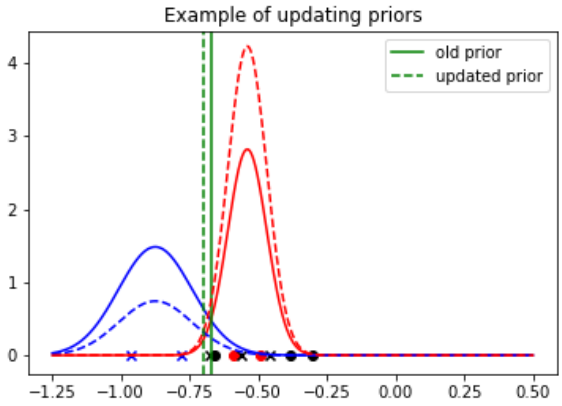


Figure 3.3: An illustration of how to update priors for a classification task. The solid blue curve is the estimated Gaussian for block 1 and the solid red curve is the estimated Gaussian for block 2. Their dashed counterparts are the curves corresponding to the densities after the prior update for node 6. The nodes from block 1 are x's and the nodes from block 2 are o's. Unlabeled nodes are black. The green solid line is the relevant decision boundary from the prior estimated from the observed block membership function. The green dashed line is the decision boundary after we updated the prior using information from the ordering of the edge weight distributions.

3.4 A small example (part 2)

Consider matrix C from section 2.4.1. This time we use the spectral embedding of the unweighted adjacency matrix to estimate the latent positions. We obtain estimated positions

$$\hat{X} = [-0.780, -0.460, -0.965, -0.677, -0.559, -0.384, -0.661, -0.490, -0.303, -0.589]$$

Then, with $b(1) = b(3) = 1$ and $b(8) = b(10) = 2$ known, we estimate the Gaussian parameters to obtain $\hat{\mu_1} = -0.62$, $\hat{\sigma_1} = 0.226$, $\hat{\mu_2} = -0.52$, and $\hat{\sigma_2} = 0.198$, resulting in the densities in Figure 3.3.

To implement the newly proposed method we must first estimate the orderings for each block, which requires estimating three means. The mean of the edge weights 1) between training data within block 1; 2) between training data from block 1 and block 2; 3) between training data within block 2. In

our case we get $\bar{X}_{1,1} = 2$, $\bar{X}_{1,2} = \bar{X}_{2,1} = 3$, $\bar{X}_{2,2} = 2$ which lead to the local orderings $\hat{O}_1 = (1,2)$ and $\hat{O}_2 = (2,1)$.

Now consider the ordering associated with node 6, $\bar{O}_6 = (2,1)$. We calculate the footrule distance and add one to get $S_6 = (1/4, 3/4)$. The new class membership priors are then given by $\hat{\pi}_{6,1} = 1/4$ and $\hat{\pi}_{6,2} = 3/4$. These new priors lead to new decision boundaries (the dashed line in Figure 3.3).

3.5 Results from generated data

We look at four different settings for the two block rank one SBM with Gaussian edge weight distributions. 1) Different means and different scales; 2) Different means and same scales; 3) Same mean and different scales; 4) Same mean and same scales. Notice that for settings 1) and 2) the order assumption holds because the distributions have different means. All the simulations have $B = \begin{bmatrix} (0.52)^2 & (0.52)(0.48) \\ (0.52)(0.48) & (0.48)^2 \end{bmatrix}$, $\mathcal{F} = \begin{bmatrix} N(\mu_1, \sigma_1^2) & N(\mu_2, \sigma_2^2) \\ N(\mu_2, \sigma_2^2) & N(\mu_1, \sigma_1^2) \end{bmatrix}$, and $n \in [150, 200, 250, 300, 350, 400, 450, 500]$ where the number of training data is $\frac{n}{10}$ with $\pi_1 = \pi_2 = 0.5$.

For settings 1) and 2) $\mu_2 - \mu_1 = 2$. For settings with equal variances, $\sigma_1 = \sigma_2 = 9$. When they are not equal, $\sigma_1 = 4$ and $\sigma_2 = 9$. Networks are generated conditioned on the number of nodes and training data in each block. Figure 3.4 shows the misclassification rate versus the number of nodes in the network. Error bars represent the 95% confidence interval for the average of 100 iterations.

In the top two plots of Figure 3.4, both the new classifier (referred to as updated priors) and pass-to-ranks tend to perform better with a larger node

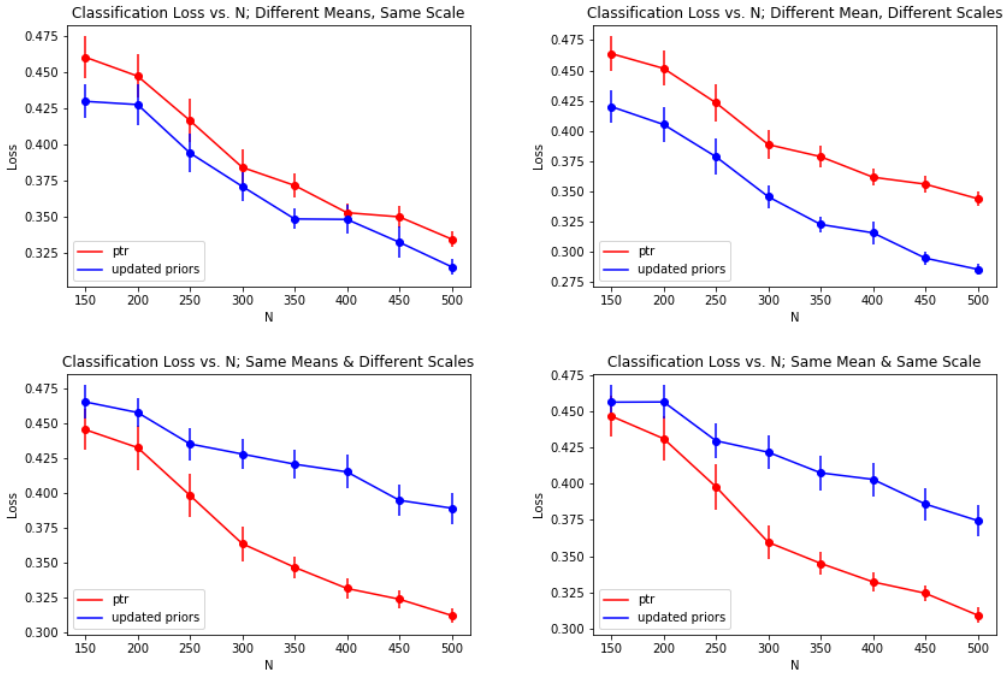


Figure 3.4: The figure on the top left plot show the results for setting 1. The top right figure show the results for setting 2. The figure on the bottom left is for setting 3 and the figure on the bottom right is for setting 4. See section 3.5 for analysis.

set. This is reassuring and can be attributed to the fact that the adjacency spectral embedding is at the core of both methods. Another reason for the similar trends in settings 1) and 2) is that pass-to-ranks and updated priors use the edge weight information in a similar way when the means are actually different. This is especially true when the variances are the same. In fact, the difference between the two plots can be attributed to the variances being equal in one setting, which pass-to-ranks can naturally take advantage of, and the variances being different in the other.

For the bottom two plots of Figure 3.4, the results are essentially flipped – pass-to-ranks outperforms updating priors. This is likely due to the fact that, while pass-to-ranks does not ignore the edge weights, it does not attempt to

use them in any explicit manner to determine class membership. In other words, when the edge weights do not encode information, or we are ill-equipped to use it, any attempt to explicitly use this non-information costs a lot in terms of misclassification. One way to address this issue is via hypothesis testing and is discussed in section 3.6.

We also consider edge weights that were generated from Poisson distributions. In particular, we consider the weight distribution matrix $\mathcal{F} = \begin{bmatrix} Pois(\mu_1) & Pois(\mu_2) \\ Pois(\mu_2) & Pois(\mu_1) \end{bmatrix}$ with the same n and B as before. We ran each simulation 100 times. The case where the order assumption does not hold again leaves some room for improvement.

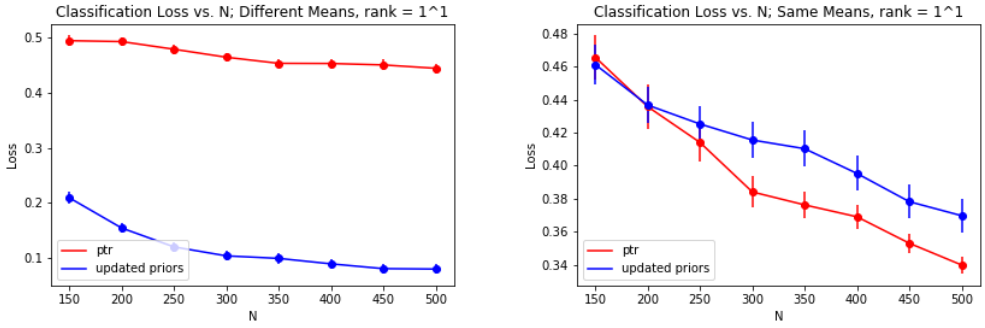


Figure 3.5: The figure on the left was generated with $\mu_1 = 3$ and $\mu_2 = 6$. The right figure was generated with $\mu_1 = \mu_2 = 3$.

3.6 Testing for a Difference in the Means

As we see in the results presented in section 3.5, the new method performs extremely well in classification tasks when the order assumption holds. The same can not be said when the assumption fails. For this method to be practical we need to check if the ordering assumption holds before proceeding

to update the priors. One way to check the assumption is through hypothesis testing. We consider the null $E(F_{u,v}) = E(F_{s,t})$ for all $u, v, s, t \in [K]$ against the alternative $E(F_{u,v}) \neq E(F_{s,t})$ for any $u, v, s, t \in [K]$. We continue to focus on the two block case.

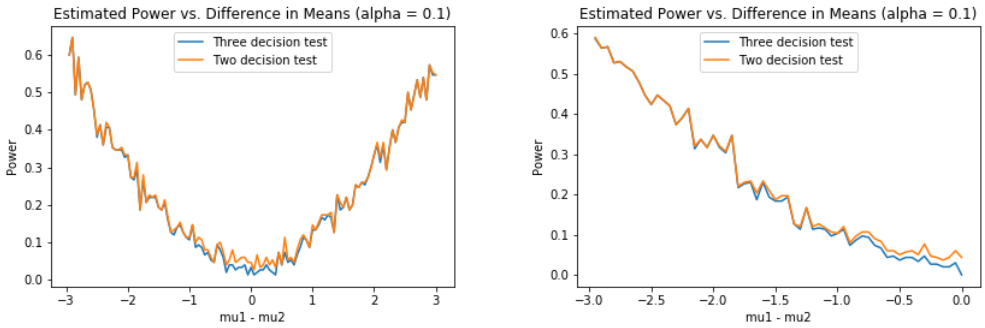


Figure 3.6: Estimated power curves the three decision test and two decision test for an SBM with $n = 200$, $\pi_1 = \pi_2 = 0.5$, number of seeds $= \frac{n}{10}$, $B = \begin{bmatrix} (0.52)^2 & (0.52)(0.48) \\ (0.52)(0.48) & (0.48)^2 \end{bmatrix}$, $\mathcal{F} = \begin{bmatrix} N(\mu_1, 16) & N(\mu_2, 16) \\ N(\mu_2, 16) & N(\mu_1, 16) \end{bmatrix}$. The plot on the left was generated via a simulation run 150 times for $\mu_2 = (\mu_1 - 3, \mu_1 + 3)$ with resolution 0.05. Since our setting is symmetric about $\mu_1 - \mu_2 = 0$, the second plot looks at $\mu_1 - \mu_2 < 0$ for 300 iterations.

Here we also care about which ordering holds, i.e. $E(F_{1,2}) < E(F_{1,1})$ or $E(F_{1,1}) < E(F_{1,2})$. We are in a testing situation where our action can take on three values. We can fail to reject the null, we can reject null and decide $E(F_{1,2}) < E(F_{1,1})$, or we can reject the null and decide $E(F_{1,1}) < E(F_{1,2})$. To perform this test in our setting we need a non-parametric test like the Mann-Whitney U (MWU) test, which tests for the equality of the locations of the distributions.

First, we calculate the p-value associated with the test statistic. If the p-value is less than some pre-selected α then we reject the null. Then, if $E(F_{1,1}) < E(F_{1,2})$ we decide that $E(F_{1,1}) < E(F_{1,2})$. Otherwise we decide that

$E(F_{1,2}) < E(F_{1,1})$. If we choose α to be large then we are more likely to reject the null and proceed to update the priors. Here the choice of α can reflect our willingness to move the decision boundaries for each node.

If we'd like to discuss how a test behaves under the null and under the two alternatives, we must first define errors in this testing scenario and, subsequently, define power. There are three types of error associated with the proposed test. Type I error, which is to incorrectly reject the null; Type 2 error, which is to incorrectly fail to reject the null; and Type 3 error, which is to correctly reject the null but incorrectly assign the order. We define power to be the probability of correctly rejecting the null and correctly ordering the distributions.

We resort to simulation to gain insight on the properties of this test in our setting. Figure 3.6 gives the power curves for the three decision test, along with the two decision test for reference. The complete simulation setting is given in the caption under the figure. It is important to point out that the three decision test has less power for μ_2 close to μ_1 but, as the difference $|\mu_1 - \mu_2|$ increases, the power curves are indistinguishable. We also note that the plot is symmetric about $\mu_1 - \mu_2 = 0$ due to the equal scale setting. While we do not correctly reject often in the settings we consider in section 3.5 (where $|\mu_1 - \mu_2| = 2$) for $\alpha = 0.1$, the selection of α is arbitrary and so it is unclear how we should interpret these results.

Incorporating the results from the test into the proposed method is simple: Update the priors if we reject the null and keep the original priors otherwise.

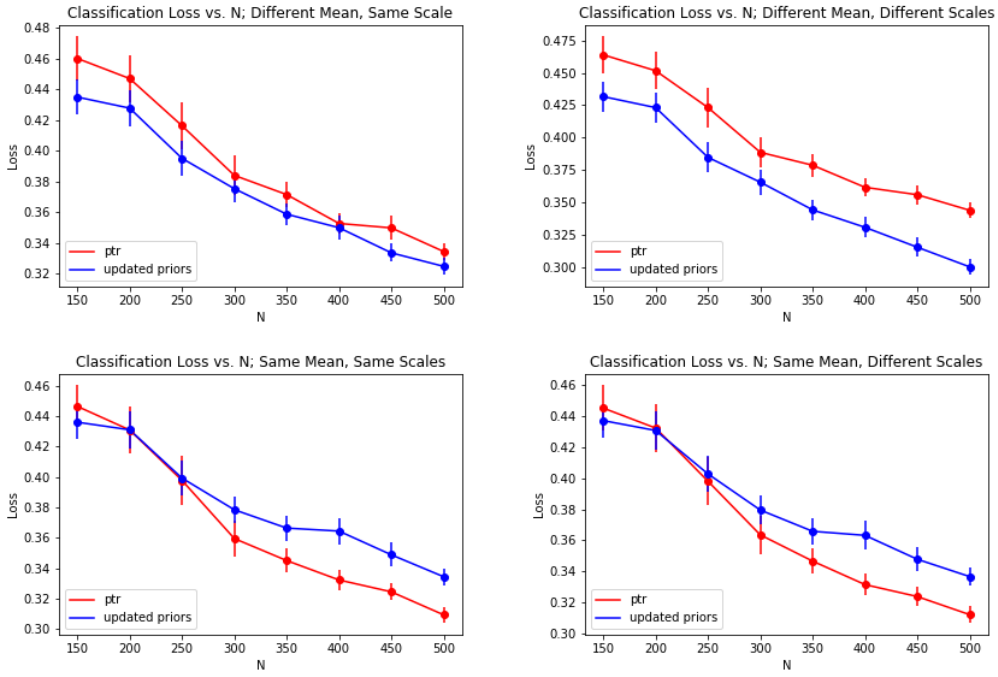


Figure 3.7: Simulation settings revisited where a hypothesis test (alpha = 0.1) for the difference in the means used to determine if we update priors. If we fail to reject we classify using the adjacency spectral embedding of the unweighted network. We can see that the testing procedure makes our procedure a bit more robust. The two charts are very similar since we fail to reject often.

In Figure 3.7 we revisit the simulation settings from before and now incorporate a hypothesis test for a difference in the means. We see from the top two plots in Figure 3.7 that our method is still preferred over pass to ranks when the order assumption holds. In the settings where the order assumption does not hold, our method is outperformed but the gap between the two methods is smaller.

3.6.1 Dynamic Additive Smoothing

We can also use the output of the test to inform the additive smoothing by changing the plus one smoothing to plus $q(\cdot)$ smoothing, where $q : [0,1] \to [1,r]$. This can be thought of as taking a p value as an input and outputting a real number between 1 and r, where $r \in \mathbb{R}$ is "large". In our setting we first have to apply a function to a collection of p values to give us a single value in $[0,1]$. In the simulation study we use Fisher's Method (see section 4.1) to combine p values. Recall that if we were to use plus r smoothing then we would essentially not update our priors (see section 3.3).

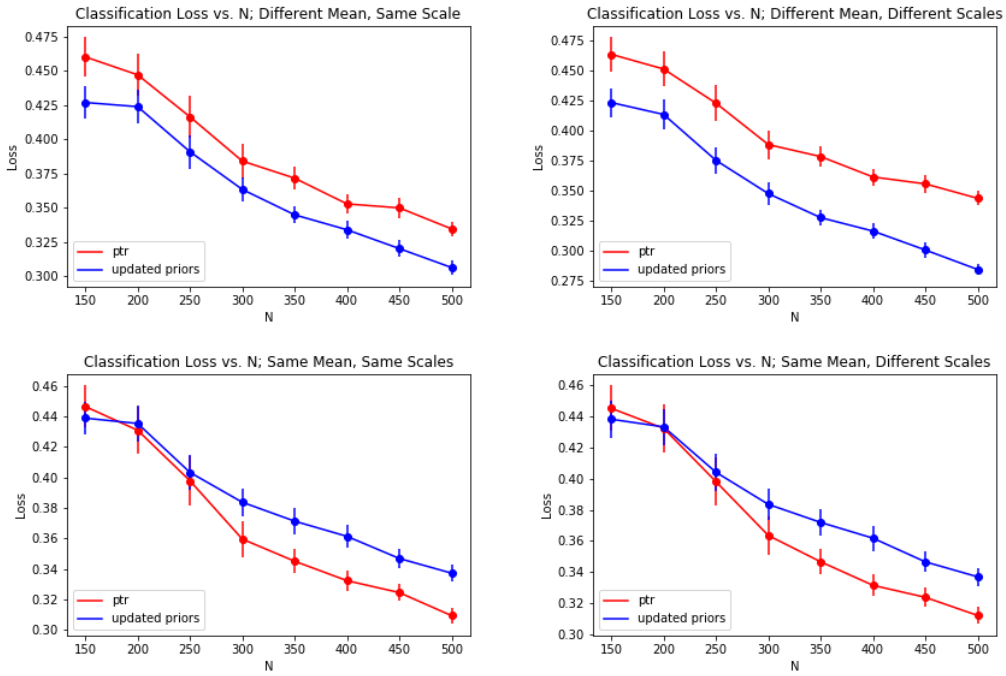


Figure 3.8: Simulation settings revisited. Dynamic additive smoothing is used to create a more robust classification procedure.

Here we are just using the fact that we can interpret a small p value as evidence against the null. We consequently inform our additive smoothing

procedure instead of operating on a binary test result. We can use additive smoothing to put us in a space that is operationally between the null and the alternative.

Figure 3.8 shows simulation results for dynamic additive smoothing, with a story similar to the results of Figure 3.7. One important distinction, however, is that the performance of pass-to-ranks and updated priors are a bit more separated in settings 1) and 2). Using dynamic additive smoothing results in improved performances for settings 3) and 4).

Chapter 4

General Edge Weight Distributions

In this section we modify the assumptions on the edge weight distributions but continue to use a measure of similarity to update priors. The methods that are proposed here are similar in spirit to the one proposed in section 3 – simply replace the S_i of section 3 with the S_i of this section to obtain updated class membership priors to use for classification.

In this section we treat the most general edge weight distribution matrix that is brought up in section 1.1, and is the motivating setting for the majority of the preceding analysis. Recall that here we are going to deal directly with the empirical cumulative distributions. We compare vectors of empirical cumulative distributions for each block and to the corresponding vector for each unlabeled node. Luckily for us, we do not need to invent the wheel for these types of comparisons and can, instead, use a transformation of the p-values from a collection of Kolmogorov-Smirnov (KS) 2-sample tests to obtain a measure of similarity and subsequently update our class membership priors.

Fisher's Method is one way to transform a collection of p values into a

single p value. The method uses the fact that $T = -2\sum_{i=1}^{K} p_i \sim \chi^2_{2K}$. This follows from applying the inverse transform method to a random variable distributed exponential(1) and then scaling it by a factor of two to obtain a χ^2 distribution with 2 degrees of freedom. Finding the p value associated with the collection of p values then comes down to calculating the "extremeness" of Fisher's T.

4.1 Methodology

We first re-introduce the notation in section 1.1. That is, we denote $\mathcal{F}_u$ as the vector of empirical cumulative distribution functions corresponding to block u and $\mathcal{F}(i)$ as the vector of empirical cumulative distribution functions corresponding to the unlabeled node i. Figure 4.1 gives some intuition into what we are looking for when we are define a similarity metric on the space of empirical distribution functions. If we were classifying solely on the information in Figure 4.1 we'd clearly label the unlabeled node as block 1. Of course, this is not the only class membership information available, so we should convert this intuition into a similarity metric and then update our priors as before.

The two-sample Kolmogorov-Smirnov test, which tests $F_1 = F_2$ against $F_1 \neq F_2$ yields a p-value that can be interpreted as a similarity metric. To make this clear, we need some notation. Let $\mathcal{F}(i)_v$ be the distribution governing the edge weights between unlabeled node i and block v. Similarly, let $\mathcal{F}_{u,v}$ be the distribution governing the edge weights between block u and block v. Since our unlabeled node is from one of the K blocks, this means that $\mathcal{F}(i)_v = \mathcal{F}_{u,v}$

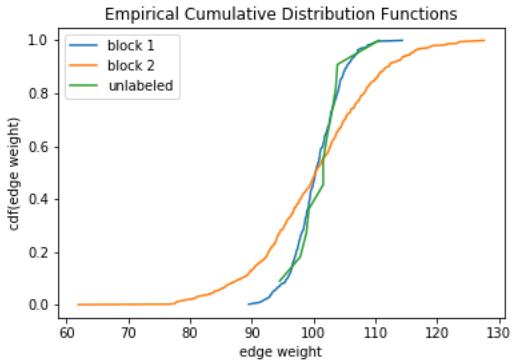


Figure 4.1: An illustration of how empirical cumulative distributions can encode class membership information.

for some u. Then a natural test to perform is $\hat{\mathcal{F}}(i)_v = \mathcal{F}_{u,v}$ against the two-sided alternative for all u. The p-value from this test can then be used as a building block for a similarity metric on this space. Holding u constant and performing this test across all v we get a collection of p values corresponding to block u. Then, combining the p-values can be done using Fisher's method, $T_{i,u} = -2\sum_{j=1}^{K} \log(p_{i,u,j}) \sim \chi^2_{2K}$ where $p_{i,u,j}$ is the p value resulting from the test $\hat{\mathcal{F}}(i)_j = \hat{\mathcal{F}}_{u,j}$. We denote the p value associated with $T_{i,u}$ as $p_{i,u}$. If we let $S_i = (p_{i,1}, \ldots, p_{i,K})$ then updating priors is as before, i.e.

$$\hat{\pi}_i = \frac{1}{\langle \pi, S_i \rangle}(\pi_1 p_{i,1}, \ldots, \pi_K p_{i,K})$$

and the resulting classifier is

$$g_G(i) = \arg\max_{u \in [K]} \hat{\pi}_{i,u} f_j(x_i | b(i) = j)$$

where G is homage to the general treatment of the edge weight distributions.

4.2 Results from generated data

For our simulation study we return to the settings in section 3. The top two plots of Figure 4.2 show the effectiveness of our proposed classifier for settings 1) and 2), which corresponds to settings where $\mu_1 \neq \mu_2$. In fact, we do not lose much compared to the order assumptions even when the scales are the same – which is the setting we'd expect the classifier built on the order assumption to do better. Our new classifier, however, clearly outperforms $g_R(\cdot)$ in setting 2). This is attributable to the fact that the KS test is able to account for a difference in scale and a difference in means.

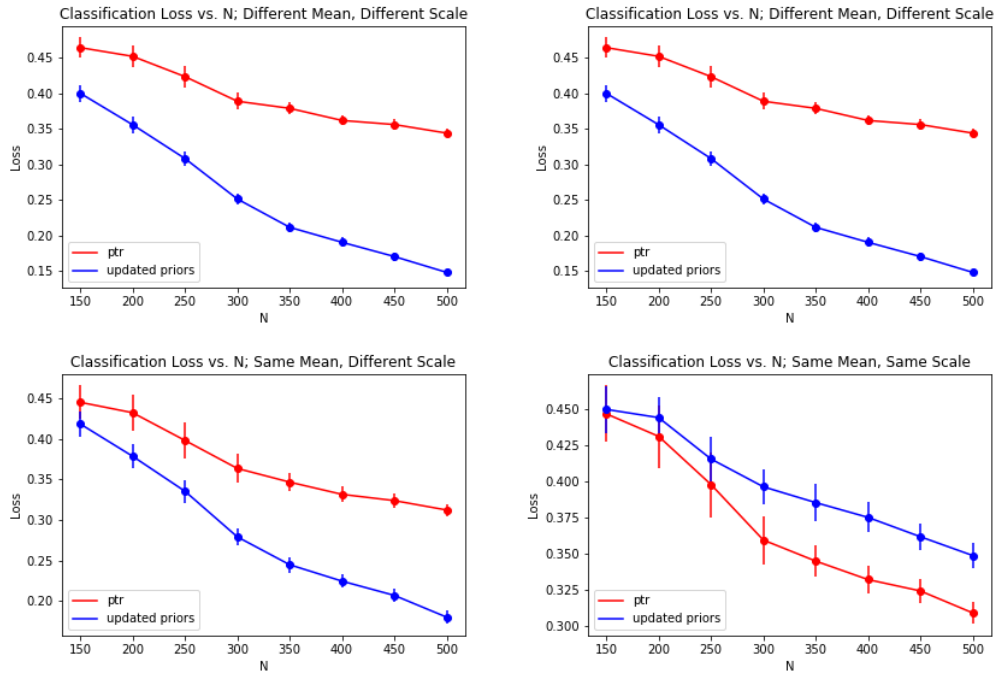


Figure 4.2: The top two figures plot the classification results for settings 1) and 2). The bottom two figures plot the classification results for settings 3) and 4).

The bottom two plots of Figure 4.2 look at settings 3) and 4), or the settings where the order assumption does not hold. We see, on the left, that $g_G(\cdot)$ is

able to outperform pass-to-ranks by accounting for scale. When there is no information in the edge weights pass-to-ranks still outperforms our classifier.

It has become clear that we are able to leverage class membership information encoded in the edge weights to create better classifiers when the edge weights actually encode class membership information. In setting 4), pass-to-ranks will continue to outperform any classifier that makes explicit assumptions on the edge weights simply because we introduce more variance into our model. We briefly mention one possible way to mitigate the effect of misspecification in the discussion below.

Chapter 5

Discussion and Conclusion

5.1 Discussion

While the methods above are effective when there is class membership information encoded in $\mathcal{F}$, we do not address all assumptions and methods.

One class of assumptions not treated here is the set of parametric assumptions. The main benefit of parametric methods in this setting is the ability to use likelihoods as a measure of similarity. Consider the case where the edge weights do not encode any class membership information (i.e. simulation setting 4). As n gets large, the plug-in distributions will converge to the true distributions. This means that if two distributions are actually the same (i.e. $F_{1,2} = F_{2,2}$) the likelihood of observing the edge weights for an unlabeled node will be approximately equivalent under the two estimated distributions. When we update the priors there will be but a small change, reflecting the similarity of the distributions. Thus, the parametric framework is more flexible than the ordering assumption presented in section 3.

An interesting approach to solve the issue of misspecification (i.e. setting

4) is to use a model selection procedure to estimate the number of unique edge weight distributions. We consider this as an alternative (and perhaps more direct) method to the "plus q(p)" smoothing presented above.

5.2 Conclusion

The preceding analysis is an introduction to the types of methods that can be used for node classification on weighted networks when we assume that the adjacency and edge weight information are conditionally independent. We showed that this class of methods can improve results for classification, as compared to pass-to-ranks, when the edge weights encode class membership information.

In particular, in section 3 we proposed an effective method for node classification when $\mathcal{F}$ can be ordered. We also presented different ways to deal with model misspecification.

In section 4 we treated general edge weight distributions and showed that the only setting in which pass-to-ranks is preferred is when the the edge weight distributions do not encode class membership.

We do not claim that this class of methods is the most effective way to use this information. We also make no claim as to how these methods would perform if the parameters governing B and $\mathcal{F}$ are related in any way. It is unclear if we would even want to stay in the spectral embedding framework.

Bibliography

Abbe, Emmanuel (2017). "Community detection and stochastic block models: recent developments". In: *arXiv preprint arXiv:1703.10146*.

Athreya, Avanti, Carey E Priebe, Minh Tang, Vince Lyzinski, David J Marchette, and Daniel L Sussman (2016). "A limit theorem for scaled eigenvectors of random dot product graphs". In: *Sankhya A* 78.1, pp. 1–18.

Athreya, Avanti, Donniell E Fishkind, Keith Levin, Vince Lyzinski, Youngser Park, Yichen Qin, Daniel L Sussman, Minh Tang, Joshua T Vogelstein, and Carey E Priebe (2017). "Statistical inference on random dot product graphs: a survey". In: *arXiv preprint arXiv:1709.05454*.

Bhagat, Smriti, Graham Cormode, and S. Muthukrishnan (2011). "Node Classification in Social Networks". In: *CoRR* abs/1101.3291. arXiv: 1101.3291. URL: http://arxiv.org/abs/1101.3291.

Fishkind, Donniell E, Vince Lyzinski, Henry Pao, Li Chen, Carey E Priebe, et al. (2015). "Vertex nomination schemes for membership prediction". In: *The Annals of Applied Statistics* 9.3, pp. 1510–1532.

Gale, William and Kenneth Church (1994). "WhatâĂŹs wrong with adding one". In: *Corpus-Based Research into Language: In honour of Jan Aarts*, pp. 189–200.

Sussman, Daniel L, Minh Tang, and Carey E Priebe (2014). "Consistent latent position estimation and vertex classification for random dot product graphs". In: *IEEE transactions on pattern analysis and machine intelligence* 36.1, pp. 48–57.

Von Luxburg, Ulrike (2007). "A tutorial on spectral clustering". In: *Statistics and computing* 17.4, pp. 395–416.

Young, Stephen J and Edward R Scheinerman (2007). "Random dot product graph models for social networks". In: *International Workshop on Algorithms and Models for the Web-Graph*. Springer, pp. 138–149.

Zhu, Mu and Ali Ghodsi (2006). "Automatic dimensionality selection from the scree plot via the use of profile likelihood". In: *Computational Statistics & Data Analysis* 51.2, pp. 918–930.

Curriculum Vitae

Hayden Helm was born in Richmond, Virginia on March 20, 1997. He graduated from Salem High School in 2015 and subsequently enrolled at Johns Hopkins University. While at Hopkins he studied Applied Mathematics and Statistics with a focus in Probability and Statistics for his Bachelor's of Science degree. He enrolled in the combined Bachelor's and Master's program in Applied Mathematics and Statistics where he focused on Statistics and Statistical Learning. He earned both his Bachelor's and Master's degrees in December 2018. He currently researches pattern recognition and machine learning techniques.